第 1 章

面向对象程序设计概述

学习目标

- 理解面向过程和面向对象的概念。
- 掌握面向对象程序的核心概念——对象和类，以及它们之间的区别和联系。
- 掌握声明类、属性和方法的办法。
- 掌握创建对象和访问对象成员的方法。
- 掌握构造器的声明和使用方法。
- 理解类的封装的优点，并掌握如何根据封装的原则正确使用类的属性和方法。
- 理解 Java 源程序的布局和层次结构，并掌握 package 语句、import 语句的使用方法。

本章在介绍面向对象程序设计基本概念的基础上，通过一个项目的设计，学习如何根据面向对象的思想设计一个类（包括类中的属性、方法和构造器）、使用一个类（创建类的对象、访问对象成员）。其中，封装是一个重要的面向对象程序设计特征，是设计类的一个基本思想。

对于实际项目的开发，如何组织众多的 Java 文件（包括源文件和类文件）是一个很重要的问题，它关系到各个项目的开发是否能够顺利进行。1.3 节将对这个问题进行讨论。

本章的内容是面向对象程序设计的基础，也是学习后面章节的基础。如果读者已经具备面向对象程序设计的基本知识，本章内容可以适当跳过。

1.1 面向过程和面向对象

在软件开发过程中，存在面向过程（Process-Oriented，PO）和面向对象（Object-Oriented，OO）两种程序开发方法，相应地，我们将计算机语言分为面向过程程序设计语

言和面向对象程序设计语言两种。

1.1.1 面向过程程序设计语言

在 C 语言和其他面向过程的程序设计语言中，编程趋向于面向动作，程序员关心的是程序功能的实现。因此，C 程序员专注于编写函数。完成特定功能的几组动作组成函数，函数最终又组成了程序，函数是 C 语言的编程单元。在 C 语言中，数据当然是重要的，但是，数据的作用主要是支持函数执行的动作，完成函数的功能。面向过程的程序设计强调的是模块化、结构化的程序设计方法——通过函数实现。

1.1.2 面向对象程序设计语言

Java 是一种完全面向对象的程序设计语言。面向对象的编程思想力图使计算机语言对事务的描述与现实世界中该事务的本来面目尽可能地一致，这样可以增强程序的可理解性，并使得程序在稳定性、可维护性和可重用性方面更有优势。面向对象是对现实世界模型的自然体现和延伸，它认为现实世界的各种实体（甚至是虚构的事务）都可以看成对象，对象之间通过消息相互作用。

我们来看一个生活中的例子。当程序中要处理人员的生日信息时，如果采用 C 语言这样的面向过程程序设计语言时，可能需要为每个人定义三个整型变量（例如 int year1，month1，day1），用于保存他的生日信息。如果要保存 n 个人的生日信息，可能需要使用 n×3 个数据（即使使用数组，也需要这么多个数组元素）。值得一提的是，这些数据本身不能体现相互之间的联系——到底哪三个数据组合在一起表示一个人的生日信息，而只能靠程序员硬性规定并记录下来。就算使用了结构体，把一个人的出生年、月、日作为结构体的成员保存在一起，也无法将对生日信息的处理（例如将某人的生日日期改为 1980 年 1 月 1 日）和生日信息有机地结合起来。由此可以看到，使用面向过程的程序设计语言设计程序时，不能很好地体现现实世界中事务的本来面目，丢失了数据本来应该携带的一些逻辑信息，增加了程序维护和重用的难度。而如果采用面向对象的程序设计语言来设计，则可以把一个人的生日信息看成一个对象，其中包括生日完整的年、月、日信息，更重要的是，这个对象中还包括对有关数据的操作功能，并且能够在数据操作时增加必要的校验和控制功能（例如可以校验所设置的日期是否是一个合法的日期），这样就大大简化了处理和传递数据的复杂程度。

从上面的例子可以看出，一个对象需要由两部分来描述：第一部分是它的“数据”，包括该对象的特性、状态等静态信息；第二部分是它的“行为”，包括对该对象的操作、对象的功能等能动信息。数据和行为是对象的两个基本特征。

例如，现实世界中的一辆汽车就可以看作一个对象，它可以行驶，这就是它的行为，另外，汽车的品牌、载客的数量、行驶公里数、行驶状态，都是它的特性和状态信息，也就是它的数据。与“汽车”相对应的是“司机”，一位司机也可以看成一个对象，司机可以驾驶汽

车，这是司机的行为；另外，司机具有人的所有属性（例如姓名、性别、年龄等），这些是司机这个对象的数据。“汽车”和“司机”这两个对象是通过行驶和驾驶这样的行为相互作用的。因为司机有驾驶汽车的技能，而汽车有行驶的本领。所以当司机在驾驶的行为中发出行驶的命令以后，汽车的行驶状态就变成“行驶中”，行驶公里数也开始逐渐增加。这样，就在对象和对象之间产生了联系。

因此，我们说，面向对象程序员关注的焦点是对象而非函数（在 Java 中称为方法），包括对象中的数据和对数据的操作——行为。

1.1.3　面向过程和面向对象程序设计

程序设计的根本任务是通过合适的计算机语言描述问题和解决问题。在以前学过的程序设计基础相关的课程中，我们学过“程序＝数据结构＋算法”这个著名的公式。其中，数据结构是用来描述问题的，而算法是用来解决问题的，这是面向过程程序设计的基本思想，如图 1-1 所示。

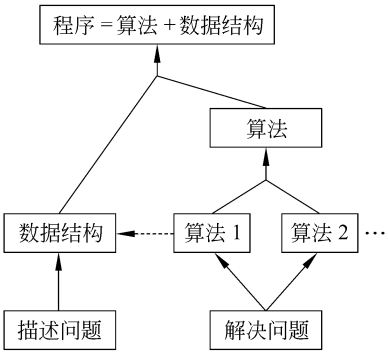

图 1-1　面向过程程序设计示意图

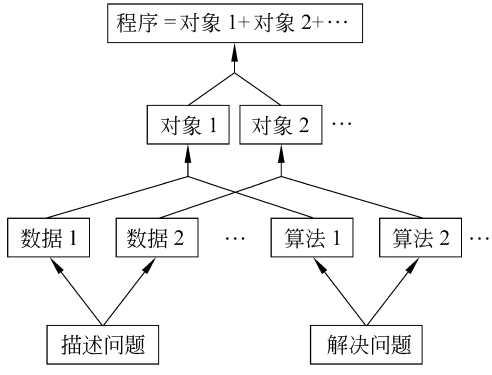

图 1-2　面向对象程序设计示意图

而面向对象的程序设计，其程序是由一个个对象组成的，对象包括数据以及作用于这些数据上的操作两个部分。因此，我们可以把面向对象的程序设计表示为如图 1-2 所示。

1.2　类

【项目 1-1】 某城市正在进行人口普查工作，需要对该城市所有人员的出生年月日信息进行记录，同时要求工作人员应该能够随时读取、修改某个人的出生年月日，并将其显示在屏幕上。

项目分析

根据【项目 1-1】的要求，我们可以按功能将其划分成以下三个模块。

模块一：记录被普查人员的出生年月日信息(设计类)；

模块二：为工作人员提供读取、修改某个人的出生信息的接口(设计类的方法)；

模块三：工作人员能够将某个人的生日信息显示在屏幕上(类的使用)。

要完成这三个功能模块的要求，需要用到类的相关知识。

1.2.1 对象和类

对于现实世界中存在的类似的多个实体，我们可以将其抽象概括为某一类事务。类似地，为了减少麻烦，对于在数据和行为上类似的多个对象(object)，可以将其抽象为一个类(class)来表示。类是对象的抽象。

正如建筑设计师可以先绘制出建筑物的图纸，然后再根据这个图纸(蓝图)建出很多相同的建筑物一样，类是对象的蓝图，可以根据实际的需要实例化出多个相互独立的对象。又例如，“人”是一个类，它描述了能够直立行走、能使用语言和生活工具、具有思维的一类高级动物，它是一个抽象的概念。而“张三”是“人”这个类的一个具体存在的对象，是实实在在的一个人，例如张三是一个高 170cm、黑眼睛、黄皮肤、25 岁的一个男人，是“人”这个类的一个实例(instance)。从这个意义上说，类(抽象的)是对象(具体的)的模板，而对象是类的实例。

类具有对象的两个要素：一是描述对象的数据元素，称为类的属性(attribute)；另外一个是描述对象的行为和功能，称为类的方法(method)，它用于操作对象或执行相关对象之间的交互。属性和方法一起称为类的成员(member)。

为了完成【项目 1-1】，可以将被普查人员的生日信息看成一个类——BirthDate，而具体某个人的生日信息看成一个具体的对象。在 BirthDate 类中，可以包括用于描述生日的年、月、日信息的 year、month、day 三个属性，以及设置和获取 year、month、day 三个属性的六个方法。程序可以设计如下：

程序 1-1 设计记录生日信息的 BirthDate 类。

目的：理解类、属性和方法的概念。

分析：该程序设计了记录生日信息的 BirthDate 类，该类声明三个属性，分别用于表示生日的年、月、日信息，并提供三个设置属性和三个读取属性的方法。

```
/**********************************************************/
/* 程序名：BirthDate.java                                  */
/* 功能：设计记录生日信息的 BirthDate 类                    */
/* 作者：池瑞楠                                            */
/* 日期：2005-04-20                                        */
/**********************************************************/
public class BirthDate {
    /* 声明三个属性 */
    private int year;
```

```
    private int month;
    private int day;

    /* 声明三个设置属性的方法 */
    public void setYear(int newYear) {
        year = newYear;
    }
    public void setMonth(int newMonth) {
        month = newMonth;
    }
    public void setDay(int newDay) {
        day = newDay;
    }

    /* 声明三个获取属性的方法 */
    public int getYear() {
        return year;
    }
    public int getMonth() {
        return month;
    }
    public int getDay() {
        return day;
    }
}
```

下面，结合程序 1-1 归纳一下声明类、声明属性和声明方法的方法。

1.2.2　类的声明

在 Java 中，要使用类必须首先声明一个类。类的声明也称为定义，其基本的语法格式如下：

```
[<修饰符>] class <类名> {
    [<属性声明>]
    [<构造器声明>]
    [<方法声明>]
}
```

说明：<>里面是描述条款；[]代表可选项；<类名>是类的名字，可以是任何合法的标识符。

编程规范

我们提倡类的命名应该做到“见名知义”，而且通常类名应该是名词，第一个字母和名字内其他所有单词的第一个字母应该大写，其他字母小写。例如：BirthDate、Custom、

Order。

＜修饰符＞可以使用 public、abstract 和 final 等关键字，现在我们只使用 public，用该关键字声明的类可以被任意访问。其他的修饰符的用法在后面的学习中将逐个介绍，本章不做解释。

在类声明的正文中还可以声明与类相关的属性、构造器和方法的集合。

1.2.3 属性的声明

类声明中所包括的数据称为属性，例如程序 1-1 中的 year、month 和 day 就是类 BirthDate 的属性。属性声明的基本语法格式如下：

```
[＜修饰符＞] ＜类型＞ ＜属性名称＞
```

- ＜修饰符＞可以使用 public、private、protected、final、static 等，现在我们只使用 public 和 private，private 指明该属性只能被该类的方法访问。
- ＜类型＞可以是任何 Java 合法的数据类型（包括基本类型和任何类）。
- ＜属性名称＞是程序员为属性起的名称，可以是任何合法的标识符。

1.2.4 方法的声明

Java 语言中方法的声明和 C 语言中函数的声明格式类似，其基本语法格式如下：

```
＜修饰符＞ ＜返回值类型＞ ＜方法名称＞ ([＜参数列表＞]) {
    [＜语句＞]
}
```

- ＜修饰符＞的使用和属性声明中修饰符的使用相同。
- ＜返回值类型＞用于说明该方法返回值的类型，它可以是任何 Java 合法的数据类型。如果方法没有返回值，一定要声明为 void。
- ＜方法名称＞可以是任何合法的标识符。

编程规范

我们提倡属性和方法的命名也应该做到“见名知义”，属性名通常为名词，而方法名通常为动词或动宾词组。第一个单词全部小写，如果有其他单词，则后面每一个单词的第一个字母大写，其他字母小写。

＜参数列表＞指明传递给该方法的参数的类型和名称。如果有多个参数，中间用逗号(“,”)隔开。

＜语句＞是该方法所完成的功能的描述，可以是零行或多行 Java 语句。如果方法需要返回值，则需要用 return 语句返回一个指明类型的值。

在上面的程序 1-1 中，声明了一个公有类，类名为 BirthDate。在类 BirthDate 中，声明了三个 int 类型的属性 year、month 和 day；三个 setXxx 方法分别用传给方法的参数值

修改对应的属性值，它们都没有返回值，将它们称为设置器；而三个 getXxx 方法分别返回三个属性的值，它们都没有参数，将它们成为读取器。

1.2.5　类的使用

正确地声明了类以及它的属性和方法以后，就可以在应用程序中使用该类了。要使用一个类，一般的做法是先以该类为模板创建一个该类的对象——实例化，然后再访问该对象的成员。

在 Java 中，使用“new 构造器”的方式创建对象，使用“对象名.对象成员”的方式访问对象成员。

对程序 1-1 中的 BirthDate 类，可以如程序 1-2 这样使用。

程序 1-2　使用 BirthDate 类，以便工作人员读取、修改某个人的生日信息。

目的：掌握创建对象和访问对象成员的方法。

分析：该程序创建了 BirthDate 类的一个对象，并使用该对象进行如下操作：使用设置器设置对象属性的值、使用读取器读取对象属性的值，这样工作人员就可以读取、修改某个人的生日信息，并将其显示在屏幕上。

```
/**********************************************************/
/* 程序名：TestBirthDate.java                              */
/* 功能：创建 BirthDate 类对象并访问其对象成员              */
/* 作者：池瑞楠                                            */
/* 日期：2005-04-20                                        */
/**********************************************************/
public class TestBirthDate {
    public static void main(String args[]) {
        /* 创建对象 birth */
        BirthDate birth = new BirthDate();

        /* 使用设置器设置对象属性的值 */
        birth.setYear(1980);
        birth.setMonth(1);
        birth.setDay(1);

        /* 使用读取器读取对象属性的值 */
        System.out.println("the birthday is " + birth.getYear() + "-"
                + birth.getMonth() + "-"
                + birth.getDay());
    }
}
```

(1) 程序运行结果

程序 1-2 的运行结果如图 1-3 所示。

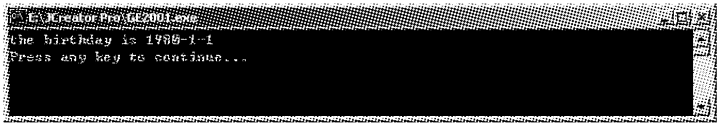

图 1-3　程序 1-2 的运行结果

(2) 程序说明

该程序中，在 TestBirthDate 的 main 方法内首先创建一个 BirthDate 类的对象 birth。然后通过如下三行代码让对象 birth 执行它的 setXxx 方法。

```
birth. setYear(1980);
birth. setMonth(1);
birth. setDay(1);
```

通过这样的“点操作”可以访问类的非私有属性和方法。

注意：如果访问本类的成员，则不需要使用“点操作”，直接使用方法名就可以了。例如，在程序 1-1 中，在 BirthDate 类的 setYear 方法中，可以不用“点操作”直接访问本类的成员 year。

1.2.6　构造器

1. 构造器的声明和调用

我们在 1.2.2 小节类的声明格式中已经看到了，可以为类声明一个构造器。构造器是 Java 类中一种特殊的方法，它用于创建该类的一个新对象。例如，当我们在程序 1-2 中使用下面的语句创建一个新对象 birth 时：

```
BirthDate birth = new BirthDate();
```

其实就是调用了 BirthDate 类的构造器创建了 birth 对象。换句话说，对象是构造的，不调用构造器，就不能创建一个新对象。构造器是一段无论何时使用 new 关键字都要运行的代码。

一个类可以声明一个或者多个构造器。通常在构造器中进行一些初始化工作，例如初始化类的实例变量的状态。

下面来看一个具体的例子：对程序 1-1 进行改造，删除其中暂时没有用到的代码，再增加一个没有参数的构造器和一个带有三个参数的构造器。这样，在程序中就可以分别调用两个构造器，创建两个不同的 BirthDate 对象。具体见程序 1-3。

程序 1-3　对不同的被普查人员，创建不同的 BirthDate 对象。

目的：掌握声明和调用构造器的方法。

分析：该程序声明多个构造器，并根据不同的被普查人员创建不同的 BirthDate

对象。

```
/**********************************************************/
/* 程序名：BirthDate.java                                  */
/* 功能：根据不同人员创建不同的 BirthDate 对象              */
/* 作者：池瑞楠                                            */
/* 日期：2005-04-20                                        */
/**********************************************************/
public class BirthDate {
    /* 声明三个属性 */
    private int year;
    private int month;
    private int day;

    /* 声明一个没有参数的构造器 */
    public BirthDate() {
        year = 2000;
        month = 1;
        day = 1;
    }

    /* 声明一个带有三个参数的构造器 */
    public BirthDate(int initYear, int initMonth, int initDay) {
        year = initYear;
        month = initMonth;
        day = initDay;
    }

    /* 用于显示属性值的一般方法 */
    public void display() {
        System.out.println("the birthday is " + year + "-" + month + "-" + day);
    }

    /* main 方法 */
    public static void main(String args[]) {
        /* 调用第一个构造器创建 birth1 对象 */
        BirthDate birth1 = new BirthDate();
        /* 调用第二个构造器创建 birth2 对象 */
        BirthDate birth2 = new BirthDate(1980, 1, 1);
        /* 分别显示两个对象的信息 */
        birth1.display();
        birth2.display();
    }
}
```

(1) 程序运行结果

程序 1-3 的运行结果如图 1-4 所示。

图 1-4 程序 1-3 的运行结果

(2) 归纳提高

① 在类中声明构造器的基本语法格式如下：

```
<修饰符> <类名> ([<参数列表>]) {
    [<语句>]
}
```

从这里可以看出，构造器和一般的方法有两点不同：

- 构造器没有返回值，而一般的方法必须指明返回值，即使是 void。
- 构造器的名称和类名相同，而一般的方法名称则没有这个限制(当然也可以和类名相同)。

② 结合上面的程序 1-3，可以看出程序中使用的两个构造器的区别如下。

- 在形式上，第一个构造器是没有参数的构造器，使用 new 关键字调用时不必提供参数，代码如下：

  ```
  BirthDate birth1 = new BirthDate();
  ```

  而第二个构造器声明时带了三个 int 型的参数，使用 new 关键字调用时必须提供参数，并且参数的类型和个数必须和声明相匹配，代码如下：

  ```
  BirthDate birth2 = new BirthDate(1980, 1, 1);
  ```

- 在功能上，两个构造器都可以创建 BirthDate 类的新对象。第一个构造器在被调用时，所创建的新对象的属性都被赋值为事先确定的值：2000、1、1；而第二个构造器在被调用时，可以根据不同的需要为所创建的新对象的属性赋予不同的值，例如程序 1-3 中被赋予了 1980、1、1。

因此，我们在程序开发过程中，可以根据实际需要声明并使用适当的构造器形式，以创建不同的类对象。

2. 默认构造器

读者可能会问，在程序 1-1 中，并没有声明构造器，那么在程序 1-2 中为什么可以使用 BirthDate birth = new BirthDate()创建 birth 对象呢？